Fair and Efficient Online Allocations with Normalized Valuations

Vasilis Gkatzelis,[1] **Alexandros Psomas,** [2] **Xizhi Tan** [1]
[1] Drexel University
[2] Purdue University
gkatz@drexel.edu, apsomas@cs.purdue.edu, xizhi@drexel.edu

Abstract

A set of divisible resources becomes available over a sequence of rounds and needs to be allocated immediately and irrevocably. Our goal is to distribute these resources to maximize fairness and efficiency. Achieving any non-trivial guarantees in an adversarial setting is impossible. However, we show that normalizing the agent values, a very common assumption in fair division, allows us to escape this impossibility. Our main result is an online algorithm for the case of two agents that ensures the outcome is fair while guaranteeing 91.6% of the optimal social welfare. We also show that this is near-optimal: there is no fair algorithm that guarantees more than 93.3% of the optimal social welfare.

Introduction

We consider a basic problem in online fair division: a set of divisible items become available over a sequence of T rounds (one item per round), and in each round we need to make an irrevocable decision regarding how to distribute the corresponding item among a set N of n agents. The value v_{it} of each agent i for the item in round t is revealed at the beginning of that round and our goal is to ensure that the overall allocation at the end of the T rounds is fair and efficient, despite the information limitations that we face.

Prior work on online resource allocation problems such as the one above has mostly focused on maximizing efficiency. In our setting, this could easily be achieved by fully allocating the item of each round t to the agent i with the largest v_{it} value. However, this approach can often lead to outcomes that are patently unfair, which is unacceptable in many important real-world applications. For example, ensuring that the outcome is fair is crucial for food banks that allocate food each day to soup kitchens and other local charities depending on the demand (Prendergast 2017), or software engineering companies that distribute shared computational resources among their employees (Gorokh et al. 2020).

Achieving fairness in such an online setting can be significantly more complicated than just maximizing efficiency. This is mostly due to the fact that reaching a fair outcome may require a more holistic view of the instance at hand. For example, the *fair-share* property (also referred to as proportionality in some contexts), one of the classic notions of fairness, requires that each of the n agents should eventually receive at least a $1/n$ fraction of their total value for all the T items. But, agents who only value highly demanded items are harder to satisfy than agents who value items of low demand, and online algorithms may be unable to distinguish between these two types of agents soon enough. As a result, designing efficient online algorithms that also satisfy the fair-share property is an important, yet non-trivial, task.

In fact, it is easy to show that without imposing any normalization on the agent values, essentially the only algorithm that guarantees the fair-share property is the naive one that equally splits every item among all agents (see the full version of this paper for a proof). This yields an outcome that is inefficient unless all agents happen to have the same values. But, the standard approach in fair division is to normalize the agents' values so that they add up to the same constant (that constant is usually 1) (Aziz 2019). As we show in this paper, this normalization is sufficient for us to escape the strong impossibility result and achieve non-trivial efficiency guarantees while satisfying the fair-share property.

Our Results and Techniques

With the exception of a few results in Section , all of our results focus on instances involving two agents, which already pose several non-trivial obstacles.

We first consider the performance of non-adaptive online algorithms, i.e., algorithms whose allocation decision in each round t depends only on the agents' values for item t. A major benefit of these algorithms is that they need not keep track of any additional information, making them easy to implement. We focus on the interesting family of *poly-proportional* algorithms that are parameterized by a value $p \geq 0$, and in each round t allocate to each agent i a fraction of the item equal to $\frac{v_{it}^p}{\sum_{j \in N} v_{jt}^p}$. For $p = 0$, we recover the algorithm that splits each item equally among the agents (which satisfies fair-share but can be inefficient), while for $p = \infty$ we get the algorithm that allocates each item to the agent with the highest value (which is efficient but violates fair-share). Another well-studied algorithm from this family, that is used widely in practice, is the *proportional allocation* (or just proportional) algorithm, which corresponds to the case $p = 1$. We show that this algorithm satisfies fair-share and is a significant improvement in terms of efficiency: it

guarantees 82.8% of the optimal social welfare (Theorem 1).

As the value of the parameter p grows, the corresponding poly-proportional algorithm allocates each item more "aggressively", i.e., a larger fraction goes to the agents with the highest values. As a result, higher values of p lead to increased efficiency, but may also lead to the violation of the fair-share property. We precisely quantify this intuition by first showing that for all $p > 2$ the corresponding poly-proportional algorithm does not satisfy fair-share (Lemma 1). Then, we show that the poly-proportional algorithm with parameter $p = 2$, the *quadratic-proportional* algorithm, satisfies fair-share and guarantees 89.4% of the optimal social welfare (Theorem 2). As a result, we conclude that 89.4% is the optimal approximation achievable by a poly-proportional algorithm that satisfies fair-share.

Moving beyond non-adaptive algorithms, we then study the extent to which adaptivity could lead to even better approximation guarantees. With that goal in mind, we propose the family of *guarded poly-proportional* algorithms, which are a slight modification of the poly-proportional algorithm, also parameterized by p. We show that every algorithm in this family satisfies fair-share, and our main result is that the guarded poly-proportional algorithm with $p = 2.7$ guarantees 91.6% of the optimal social welfare (Theorem 3). On the other hand, we prove that no fair-share algorithm (adaptive or non-adaptive) can achieve an approximation to the optimal welfare better than 93.3% (Theorem 4), thus establishing that our positive result is near-optimal.

To prove our results, we leverage the fact that our algorithms have a closed-form expression for the agents' allocations and utilities. We can use this fact and write a mathematical program that computes the worst-case approximation to the optimal welfare over all instances. We use variables v_t for the value of agent 1 for item t and λ_t for the ratio between agents' values. Even though this program is not itself convex (so at first glance it's unclear how useful it is), we show that under a suitable choice of variables and constraints, fixing some of the variables (i.e. treating them as constants) gives a linear program with respect to the remaining variables. The majority of the constraints in this LP are non-negativity constraints, so, using the fundamental theorem of linear programming we conclude that the worst-case instance only has a few (two or three depending on the algorithm) items with positive valuations. Once we have such small instances we can analyze the approximation using simple calculus. See the proofs of Theorem 1, 2 and 3 for details.

We conclude with a brief discussion regarding instances with $n \geq 3$ agents. We already know from the work of (Caragiannis et al. 2012) on the *price of fairness* that even offline algorithms cannot achieve an approximation better than $\Omega(1/\sqrt{n})$; we complement this result by showing that the non-adaptive proportional algorithm matches this bound. Finally, we provide an interesting local characterization of all online algorithms that satisfy the fair-share property.

Related Work

The same model that we consider in this setting, i.e., online allocation of divisible items with normalized agent valuations, was very recently studied by Gorokh et al. (2020). But, rather than introducing fairness as a hard constraint, as we do here, they (approximately) maximize the Nash social welfare objective. On the other hand, Bogomolnaia, Moulin, and Sandomirskiy (2019) maximize efficiency subject to fair-share constraints, as we do, but not in an adversarial setting. The agent values are stochastically generated and fairness is guaranteed only in expectation.

An additional motivation behind our assumption that the agents' values are normalized comes from systems where the users are asked to express their value using a budget of some artificial currency in the form of tokens. If a user has a high value for a good then she can use more tokens to convey this information to the algorithm. Since all users have the same budget, their values are normalized by design. A natural, and very well-studied algorithm in these systems is the *proportional algorithm*, which distributes each item in proportion to the expressed value (see, e.g., (Zhang 2005; Feldman, Lai, and Zhang 2009; Christodoulou, Sgouritsa, and Tang 2016; Brânzei, Gkatzelis, and Mehta 2017)). We provide an analysis of this algorithm, but we also achieve improved results using alternative algorithms.

Zeng and Psomas (2020) considered the trade-off between fairness and efficiency under a variety of adversaries, but in a setting with indivisible items and non-normalized valuations. Against the strong adversary studied here, their results are negative: no algorithm with non-trivial fairness guarantees can Pareto-dominate a uniformly random allocation.

More broadly, our paper is part of the growing literature on online, or dynamic, fair division. Much of this prior work analyzes settings where the agents are static and the resources arrive over time, like we do (Benade et al. 2018; He et al. 2019). Another line of work studies the allocation of static resources among dynamically arriving and departing agents (Walsh 2011; Kash, Procaccia, and Shah 2014; Friedman, Psomas, and Vardi 2015, 2017; Im et al. 2020).

Preliminaries

We consider the problem of allocating T divisible items among a set N of n agents. A fractional allocation $\mathbf{x}$ defines for each agent $i \in N$ and item t the fraction x_{it} of that item that the agent will receive. A feasible allocation satisfies $\sum_{i \in N} x_{it} \leq 1$ for all items t. We assume the valuations of the agents are *additive*: each agent i has valuation v_{it} for item t, and utility $u_i(\mathbf{x}) = \sum_{t \in [T]} v_{it} x_{it}$ for an allocation $\mathbf{x}$. We also assume that the agents' valuations are normalized so that $\sum_{t \in [T]} v_{it} = 1$. We evaluate the efficiency of an allocation $\mathbf{x}$ using the *social welfare* (SW), i.e., the sum of all agents' utilities $SW(\mathbf{x}) = \sum_{i \in N} u_i(\mathbf{x})$.

An allocation $\mathbf{x}$ satisfies fair-share if $u_i(\mathbf{x}) \geq \frac{1}{n}$ for every agent $i \in N$. We say that an algorithm satisfies fair-share if it always outputs an allocation that satisfies fair-share. Another popular definition of fairness is envy-freeness, which dictates that no agent i values the allocation of some other agent j more than her own. It is well known that if every item t is fully allocated, i.e., $\sum_{i \in N} x_{it} = 1$, then envy-freeness implies fair-share, and for two-agent instances (which is the main focus of this paper) the two notions coincide.

The item valuations are not available to us up-front; in-

stead, the items arrive online (one per round) and the agent values for the item of round t are revealed when the item arrives. The algorithm then makes an irrevocable decision about how to allocate the item before moving on to the next round. We evaluate our algorithms using worst-case analysis, so one can think of the values being chosen by an adaptive adversary aiming to hurt the algorithm's performance. Throughout the paper, our algorithms do not need to know T, the total number of rounds, but all our inapproximability results apply even to algorithms that have this information.

We say an algorithm is *non-adaptive* if its allocation decision for round t solely depends on the valuations at round t, whereas an *adaptive* algorithm can use the valuations and allocations of all the previous rounds. An interesting family of non-adaptive algorithms parameterized by a value p are ones that we call *poly-proportional* algorithms whose allocation in each round t is proportional to v_{it}^p, i.e., each agent i is allocated a fraction $x_{it} = v_{it}^p / \sum_{j \in N} v_{jt}^p$. For $p = 0$ this become the equal-split algorithm, for $p = 1$ the proportional algorithm, and for $p = \infty$ the greedy one.

Given some algorithm $\mathcal{A}$, let $\mathbf{x}^{\mathcal{A}}(\mathbf{v})$ denote the overall allocation that it outputs on an instance with agent values $\mathbf{v}$, and let $\mathbf{x}^{\text{OPT}}(\mathbf{v})$ be the social welfare maximizing allocation. $\mathcal{A}$ is an α-approximation to the optimal social welfare if

$$\min_{\mathbf{v}} \frac{SW(\mathbf{x}^{\mathcal{A}}(\mathbf{v}))}{SW(\mathbf{x}^{\text{OPT}}(\mathbf{v}))} \geq \alpha.$$

Note that our algorithms are constrained to be online and always output fair-share outcomes, while the welfare-maximizing benchmark is restricted by neither of the two.

Non-Adaptive Algorithms

Non-adaptive algorithms have the important benefit that they need not keep track of historical information regarding the agents' allocation or preferences. A naive example of such an algorithm is equal-split, i.e,. the poly-proportional algorithm with $p = 0$. Since this algorithm splits every item equally among the two agents, they both always receive value exactly $1/2$, and hence the outcome is fair-share. However, this outcome can be very inefficient, leading to a 50% approximation to the optimal welfare (e.g., consider an instance with $v_{11} = v_{22} = 1$ and $v_{12} = v_{21} = 0$).

Our first result analyzes the widely-used proportional algorithm ($p = 1$) and shows that it guarantees 82.8% of the social welfare. This is already a big improvement compared to 50%, but we then also provide a fair-share algorithm that improves this further, to 89.4%. Proofs missing from this section can be found in the full version of this paper.

Theorem 1. *The proportional algorithm satisfies fair-share and gives a* 0.828 *approximation to the optimal welfare.*

Proof. Due to space limitations, we defer the proof of the fair-share property to the appendix in favor of including a complete sketch of the efficiency proof. Given an instance $\mathbf{v}$, let $v_t = v_{1t}$ and $\lambda_t = \frac{v_{2t}}{v_{1t}}$ for each $t \in [T]$. Let ALG be the welfare of the proportional algorithm.

$$ALG = \sum_{t \in [T]} \frac{v_t^2 + (v_t \lambda_t)^2}{v_t + v_t \lambda_t} = \sum_{t \in [T]} v_t \frac{1 + \lambda_t^2}{1 + \lambda_t}.$$

Now, consider the following mathematical program:

$$\begin{aligned} &\text{minimize } \textstyle\sum_{t \in [T]} v_t \frac{1+\lambda_t^2}{1+\lambda_t} \\ &\text{subject to } \textstyle\sum_{t \in [T]} v_t = \sum_{t \in [T]} v_t \lambda_t \qquad (1) \\ &\qquad \textstyle\sum_{t \in [T]: \lambda_t \leq 1} v_t + \sum_{t \in [T]: \lambda_t > 1} \lambda_t v_t = 1 \qquad (2) \\ &\qquad v_t, \lambda_t \geq 0, \text{ for all } t \in [T] \end{aligned}$$

The objective is to minimize the approximation to welfare we receive from the algorithm. In this program, we don't enforce that the agents' values add up to 1, but we simply have them be equal to each other (constraint 1). Instead, we ask that the optimal welfare is equal to 1 (constraint 2).

First, we argue that solving this program would return the worst-case approximation to welfare. Consider an arbitrary feasible solution $\mathbf{v}, \lambda$ to this program; dividing each agent's values (each v_{it}) by their common total value $\sum_{t \in [T]} v_{it}$ yields a feasible normalized instance of our problem. Given the constraints of the program, the optimal social welfare for this instance is equal to 1 over the normalization term $\sum_{t \in [T]} v_{it}$. Also, the social welfare of the proportional algorithm is equal to the program's objective divided by the same normalization term. Therefore, the approximation factor for this instance is equal to the value of the objective.

Second, notice that for any fixed λ, the remaining program, with variables only the v_ts, is a linear program with T variables. By the fundamental theorem of linear programming, a minimizer occurs at the region's corner, i.e. there is a minimizer with T constraints tight. Since the total number of constraints is $T + 2$, and the first two constraints are tight, $T-2$ of the T tight constraints are non-negativity constraints. So the worst-case approximation occurs when there are exactly two variables/rounds with a positive value for agent 1. Without loss of generality (the proportional algorithm is memoryless) these are the first two items.

Third, for every instance where agent 1 values only the first two items, the approximation to optimal welfare is minimized when agent 2 also values only the first two items.

Now, consider the two rounds instance, in the original notation, where agent 1 has value v_1 for item 1 and $1 - v_1$ for item 2, while agent 2 has values $1 - v_2$ and v_2. Without loss of generality $v_1 \geq 1 - v_2$, which implies $v_2 \geq 1 - v_1$. Therefore, $OPT = SW(\mathbf{x}^{\text{OPT}}(\mathbf{v})) = v_1 + v_2$, and

$$ALG = \frac{v_1^2 + (1 - v_2)^2}{v_1 + 1 - v_2} + \frac{(1 - v_1)^2 + v_2^2}{v_2 + 1 - v_1}.$$

Then, overloading notation, we have that the approximation to the welfare is

$$\alpha(v_1, v_2) = \frac{\frac{v_1^2 + (1-v_2)^2}{v_1 + 1 - v_2} + \frac{(1-v_1)^2 + v_2^2}{v_2 + 1 - v_1}}{v_1 + v_2}.$$

We analyze this function, by taking partial derivatives and analyzing all critical points. We find that the worst approximation to optimal welfare is achieved for $v_1 = v_2 = 1/\sqrt{2}$, and has value $\alpha\left(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\right) = 2(\sqrt{2} - 1) \approx 0.828$. See the full version of this paper for the missing details. □

Performance of Poly-Proportional Algorithms

We now study the family of poly-proportional algorithms more broadly. As we mentioned in the introduction, poly-proportional algorithms with higher values of p may lead to increased social welfare, but they also make it increasingly likely that the fair-share property will be violated. We first show that we cannot increase p by too much before losing fair-share: for any $p > 2$ the corresponding poly-proportional algorithm does not satisfy fair-share.

Lemma 1. *The poly-proportional algorithm with parameter p does not satisfy fair-share for any $p > 2$.*

Proof. Consider the following two item instance. The first round has values x and 1 for agents 1 and 2, respectively, while the second round has values $1 - x$ and 0. Agent 1 has utility $\frac{x \cdot x^p}{1+x^p} + 1 - x = 1 - \frac{x}{x^p+1}$. For $x = (\frac{1}{p-1})^{1/p}$, agent 1 gets utility $u_1 = 1 - \frac{p-1}{p}(\frac{1}{p-1})^{1/p}$. we have $\frac{d}{dp}u_1 = -\frac{\ln(p-1)(p-1)^{\frac{p-1}{p}}}{p^3}$, notice that for any $p > 2$, $\frac{d}{dp}u_1 < 0$. Also since when $p = 2$ agent one's utility is $1 - \frac{1}{2}(\frac{1}{1})^{1/2} = \frac{1}{2}$, agent one's utility is less than $\frac{1}{2}$ for any $p > 2$. □

Our main result in this section is for the poly-proportional algorithm with parameter $p = 2$: we call this the *quadratic-proportional* algorithm. We show that this algorithm satisfies fair-share and achieves a 0.894 approximation to the optimal welfare, a significant improvement over the proportional algorithm. By Lemma 1, the quadratic-proportional algorithm guarantees the optimal social welfare within the class of fair-share poly-proportional algorithms.

Theorem 2. *The quadratic-proportional algorithm satisfies fair-share and achieves a 0.894 approximation to the optimal social welfare.*

Theorem 2 follows from Lemmas 2 and 3.

Lemma 2. *The quadratic-proportional algorithm satisfies fair-share.*

Proof. It suffices to show that agent 1 gets utility at least $1/2$ in all instances: if this holds, then the same holds for agent 2, by symmetry. Given any instance, we first show that merging and splitting certain items (rounds) results in a new instance where agent 1 is worse off.

Merging a set S of items with values (v_{1t}, v_{2t}) creates a new item with value $(\sum_{t\in S} v_{1t}, \sum_{t\in S} v_{2t})$. A split operation on an item with values (v_1, v_2), $v_1 \geq v_2$, creates two items, with values (v_2, v_2) and $(v_1 - v_2, 0)$.

Claim 1. *Let $\mathbf{v}$ be any instance, and let $\mathbf{v}'$ be the instance where we split all items $t \in [T]$ such that $\frac{v_{1t}}{v_{2t}} > 1$, with $v_{2t} > 0$. Then the utility of agent 1, in the quadratic-proportional algorithm, in instance $\mathbf{v}'$ is at most her utility in instance $\mathbf{v}$.*

Proof. It suffices to show that the utility of agent 1 weakly decreases after splitting a single item with values $v_1 = x, v_2 = y$, such that $\frac{x}{y} \geq 1$. Let u be the utility of agent 1 (for this item) before splitting and u^* the utility after splitting. We have that $u = \frac{x^{p+1}}{x^p+y^p}$ and $u^* = \frac{y^{p+1}}{2y^p} + x - y = x - \frac{y}{2}$.

$$u - u^* = \frac{y^{p+1} - 2xy^p + yx^p}{2x^p + 2y^p}.$$

It suffices to show that this is non-negative for all $x \geq y$. Since $2x^p + 2y^p \geq 0$, we only need to show that $y^{p+1} - 2xy^p + yx^p \geq 0$. Dividing both sides by y^{p+1}, we have $1 - 2\frac{x}{y} + (\frac{x}{y})^p \geq 0$. For $p = 2$, the LHS is equal to $(x/y - 1)^2$ which is non-negative. Note that we used the fact that $x > y$ to ensure that splitting was a valid operation. □

Claim 2. *Let $\mathbf{v}$ be any instance, and let $\mathbf{v}'$ be the instance where we take two arbitrary items of $\mathbf{v}$ that satisfy $\frac{v_{1t}}{v_{2t}} \leq 1$ and merge them. Then the utility of agent 1, in the quadratic-proportional algorithm, in instance $\mathbf{v}'$ is at most her utility in instance $\mathbf{v}$.*

Proof. Let a and b be the two items we want to merge, with corresponding values v_{1a}, v_{2a}, v_{1b} and v_{2b}. We show that

$$\frac{v_{1a}^3}{v_{1a}^2 + v_{2a}^2} + \frac{v_{1b}^3}{v_{1b}^2 + v_{2b}^2} \geq \frac{(v_{1a} + v_{1b})^3}{(v_{1a} + v_{1b})^2 + (v_{2a} + v_{2b})^2}.$$

We can simplify this expression to:

$$(v_{2b}v_{1a} - v_{2a}v_{1b})^2 \left(v_{2b}^2 v_{1a} + 2v_{2b}v_{2a}(v_{1a} + v_{1b}) + v_{1b}(v_{2a}^2 - v_{1a}(v_{1a} + v_{1b}))\right) \geq 0.$$

If $v_{2b}v_{1a} - v_{2a}v_{1b} = 0$ we are done. Assume that this is not the case. It suffices to show that

$$v_{2b}^2 v_{1a} + 2v_{2b}v_{2a}(v_{1a} + v_{1b}) + v_{1b}(v_{2a}^2 - v_{1a}^2 - v_{1a}v_{1b}) \geq 0.$$

First, we are going to drop the second term of the sum. Second, since $\frac{v_{1a}}{v_{2a}} \leq 1$, we have that $v_{2a}^2 \geq v_{1a}^2$, and the third term is lower bounded by $v_{1a}v_{1b}^2$. It thus remains to show that $v_{2b}^2 v_{1a} - v_{1a}v_{1b}^2 \geq 0$, which holds since $\frac{v_{1b}}{v_{2b}} \leq 1$. □

We repeatedly apply Claims 1 and 2, until no splitting or merging is possible, to get a worst case instance for agent 1. This instance will have multiple items with zero value for agent 2 that we can simply combine into a single item. Since splitting is no longer possible, there are no items $t \in [T]$ with $v_{2t} > 0$ and $\frac{v_{1t}}{v_{2t}} > 1$. Since merging is not possible there is at most one item t with $\frac{v_{1t}}{v_{2t}} \leq 1$. Therefore, we have an instance with two items, one with both positive values (that we cannot merge) and one with zero value for agent 2. Let v be the value of agent 1 for item 1, and $1 - v$ her value for item 2. Agent 2's values are 1 and 0.

Agent 1 has utility $\frac{v^3}{v^2+1} + 1 - v = 1 - \frac{v}{v^2+1}$. It is easy to confirm that this function is minimized for $v = 1$ where it takes the value $1/2$. □

Lemma 3. *The quadratic-proportional algorithm achieves a 0.894 approximation to the optimal social welfare.*

We start by showing that two item instances are the worst case. This is, in fact, true for all algorithms in the poly-proportional family.

Claim 3. *For any p, the worst-case instance (in terms of approximation) for the poly-proportional algorithm with parameter p has at most two items.*

Proof. Similarly to Theorem 1 one can write a mathematical program with variables v_t and λ_t that computes the worst-case approximation to welfare, and then observe that for every fixed choice of λ the remaining program is in fact linear. Applying the fundamental theorem of linear programming we conclude that at most two v_t variables are non-zero. We defer the details to the full version of this paper. □

Proof of Lemma 3. Given Claim 3 we only need to consider two item instances. Let v_1 and $1 - v_2$ be the agents' values for item 1, and $1 - v_1$ and v_2 their values for item 2.

Without loss of generality, assume that $v_1 > 1 - v_2$ (and therefore $v_2 > 1 - v_1$). The optimal welfare becomes $OPT = v_1 + v_2$. Consider the performance of our algorithm:

$$ALG = \frac{v_1^3 + (1-v_2)^3}{v_1^2 + (1-v_2)^2} + \frac{(1-v_1)^3 + v_2^3}{(1-v_1)^2 + v_2^2}.$$

The approximation to welfare is

$$\alpha(v_1, v_2) = \frac{ALG}{OPT} = \frac{\frac{(1-v_1)^3 + v_2^3}{(1-v_1)^2 + v_2^2} + \frac{(1-v_2)^3 + v_1^3}{(1-v_2)^2 + v_1^2}}{v_1 + v_2}$$

In the remainder of the proof we take partial derivatives with respect to v_1 and v_2 and analyze the critical points, using numerical solvers for part of the proof. The worst extreme point is $(0.6265, 0.6265)$, which gives $\alpha(0.6265, 0.6265) > 0.894$. See the full version of this paper for details. □

Adaptive Algorithms

Moving beyond non-adaptive algorithms, in this section we consider the benefits of being adaptive. In deciding how to allocate the item of each round t, adaptive algorithms can take into consideration, e.g., the utility of each agent so far, or what portion of their total value is yet to be realized. But, what would be a useful way to leverage this information in order to achieve improved approximation guarantees?

We propose a natural way to modify the family of poly-proportional mechanisms studied in the previous section. Specifically, we use the additional information to "guard" against the violation of the fair-share property. To motivate this modification, assume that at the end of some round c during the execution of a poly-proportional with $p > 2$ the utility that one of the agents has received so far plus her value for all remaining items is exactly $1/2$, i.e.,

$$\sum_{t=1}^{c} v_{it} x_{it} + \sum_{t=c+1}^{T} v_{it} = \frac{1}{2}.$$

This would mean that, unless that agent receives all of the remaining items that she has positive value for in full, then she would not receive her fair share. We refer to this as a *critical point* and use it to define the family of *guarded poly-proportional* algorithms parameterized by p: while no agent has reached a critical point, the algorithm is identical to the corresponding non-adaptive poly-proportional one; but, if some agent reaches a critical point, then all the remaining items are fully allocated to that agent. It therefore leverages adaptivity in a simple way, by checking for critical points.

Note that a critical point may not necessarily arise only at the beginning or at the end of a round. However, it is easy to show that we can assume this is the case without loss of generality. Roughly speaking, if a critical point is reached during the execution of some round t while a fraction f of that item has been allocated, then we can divide that item into two pieces (of size f and $1 - f$), creating an instance with $T + 1$ items where the critical point is reached at the end of round t, and without affecting the outcome of the algorithm. We discuss this in more detail in the full version of this paper.

If some agent reaches a critical point then, clearly, these algorithms ensure that the agent will receive her fair share. But, this does not imply that the other agent will also receive her fair share. For this to be true, the other should have received her fair share before that critical point, because she will receive no more items. Our next result shows that, in fact, this family of algorithms always satisfies fair-share.

Lemma 4. *The guarded poly-proportional algorithm with parameter p satisfies fair-share for all $p \geq 0$.*

Proof. If there is no critical point the statement trivially holds, so assume, without loss of generality, that agent 1 reaches a critical point at round c. By definition, we have that $\sum_{t=1}^{c} v_{1t} \cdot x_{1t} + \sum_{t=c+1}^{T} v_{it} = 1/2$. By the normalization assumption, $\sum_{t=1}^{c} v_{1t} \cdot (x_{1t} + x_{2t}) + \sum_{t=c+1}^{T} v_{it} = 1$. We get $\sum_{t=1}^{c} v_{1t} \cdot x_{2t} = \sum_{t=1}^{c} v_{1t} \cdot \frac{v_{2t}^p}{v_{1t}^p + v_{2t}^p} = \frac{1}{2}$. That is, it remains to show that fair-share is satisfied for agent 2.

Similarly to the proof of Theorem 1 and Lemma 2 we will write a mathematical program with variables $v_t = v_{1t}$ and $\lambda_t = \frac{v_{2t}}{v_{1t}}$, for all $t \in [t]$. The goal of the program this time will be to find a worst-case instance with respect to agent 2, given that c is a critical point for agent 1.

Agent 1's utility of resources allocated to agent 2 can be expressed as $\sum_{t \leq c} v_t \frac{\lambda_t^p}{1+\lambda_t^p}$, while agent 2 has utility $\sum_{t \leq c} v_t \frac{\lambda_t^{p+1}}{1+\lambda_t^p}$. Consider the program

$$\begin{array}{ll} \text{minimize} & \sum_{t \leq c} v_t \frac{\lambda_t^{p+1}}{1+\lambda_t^p} \\ \text{subject to} & \sum_{t=1}^{c} v_t \frac{\lambda_t^p}{1+\lambda_t^p} = \frac{1}{2} \\ & \sum_{t=1}^{c} v_t \leq 1 \\ & \sum_{t=1}^{c} v_t \lambda_t \leq 1 \\ & v_t, \lambda_t \geq 0 \text{ for all } t \in [c] \end{array}$$

Notice that given a feasible solution to this program one can always construct a valid online allocation instance, where the guarded poly-proportional algorithm with parameter p will reach critical point c for agent 1 and agent 2's utility is exactly the objective function, and vice versa. Proving the lemma is therefore equivalent to showing that the optimal solution u_2 of this program above is at least $\frac{1}{2}$.

Consider any fixed choice for the λ_t variables: the remaining program is linear, and therefore, by the fundamental theorem of linear programming we know that there exists an optimal solution with c tight constraints (since there are c

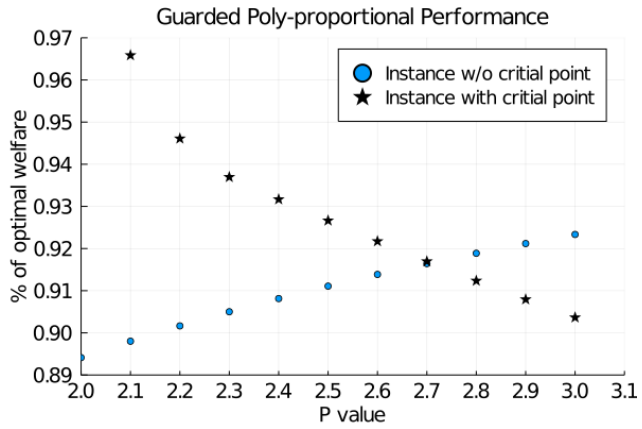

Figure 1: Approximation to the optimal welfare by guarded poly-proportional algorithms for different values of p, depending on whether the instance has a critical point or not

variables). The first constraint is already tight, so we have $c-1$ other tight constraints. At least $c-3$ of those are non-negativity constraints, so we have at most 3 positive variables. In the remainder of the proof we consider all the cases; details are deferred to the full version □

For non-adaptive algorithms, we observed that efficiency increases with p but, unfortunately, the largest value that yields a fair-share algorithm is $p=2$. For the guarded poly-proportional family we can get a fair-share algorithm for all p, but how does the efficiency depend on this value? For larger values of p, the algorithm is trying to maximize social welfare more aggressively, but this means that it is more likely to reach a critical point, after which it is forced to be inefficient. Based on a class of instances provided in the paper's full version, Figure 1 provides approximation upper bounds quantifying precisely this trade-off: if for each p we restrict our attention to instances where the corresponding poly-proportional algorithm does not reach a critical point, then the performance increases with p. But, as p increases, the set of instances with a critical point keeps growing and the greediness of the algorithm gradually hurts its efficiency.

For each value of p the points in the plot upper bound the algorithm's approximation, so the most promising choice is $p=2.7$, where the two points meet. Our main result is that the guarded poly-proportional with parameter $p=2.7$ achieves a 0.916 approximation to the optimal social welfare which, as the figure indicates, is essentially optimal within the family of guarded poly-proportional algorithms.

Theorem 3. *The guarded poly-proportional algorithm with parameter $p=2.7$ achieves a 0.916 approximation to the optimal social welfare.*

Proof. Let α be the approximation to the optimal welfare of the algorithm. We encode an instance with variables $v_t = v_{1t}$, and $\lambda_t = \frac{v_{2t}}{v_{1t}}$, for all $t \in [T]$. Let c be the critical point (if any) and without loss of generality, assume that agent 1 reaches her critical point. Agent 2's utility $\sum_{t\le c} v_t\lambda_t \cdot \frac{(v_t\lambda_t)^p}{v_t^p+(v_t\lambda_t)^p} = \sum_{t\le c} v_t \frac{\lambda_t^{p+1}}{1+\lambda_t^p}$. Agent 1's utility is $\sum_{t\le c} v_t \frac{1}{1+\lambda_t^p} + \sum_{t=c+1}^{T} v_t$. Similarly to Theorem 1 and Lemma 3 we write a mathematical program for the optimal approximation ratio:

$$
\begin{aligned}
\text{minimize} \quad & \textstyle\sum_{t=1}^{c} v_t \frac{1+\lambda_t^{p+1}}{1+\lambda_t^p} + \sum_{t=c+1}^{T} v_t \\
\text{subject to} \quad & \textstyle\sum_{t=1}^{T} v_t = 2\left(\sum_{t=1}^{c} v_t \frac{1}{1+\lambda_t^p} + \sum_{t=c+1}^{T} v_t\right) \\
& \textstyle\sum_{t=1}^{T} v_t = \sum_{t=1}^{T} v_t\lambda_t \\
& \textstyle\sum_{t\in[T]:\lambda_t\le 1} v_t + \sum_{t\in[T]:\lambda_t>1} \lambda_t v_t = 1 \\
& v_t \ge 0, \text{ for all } t \in [T] \\
& \lambda_t \ge 0, \text{ for all } t \in [T]
\end{aligned}
$$

The first constraint encodes the fact that c is a critical point: the LHS is the total value of agent 1, while the RHS is twice the utility of agent 1. These should be equal since c is a critical point for agent 1. The second constraint equalizes the agents' values (instead of normalizing them to 1), while the third constraint normalizes the optimal welfare to 1. One can go from an arbitrary feasible solution of this program to a valid instance by dividing each v_{it} by $\sum_{t=1}^{T} v_t$, and vice versa, while the approximation to the optimal welfare (which is equal to the welfare when the optimal welfare is 1) is exactly the objective of this program.

Now observe that for every fixed choice of the λ_t variables we get a linear program (with respect to the v_t variables):

$$
\begin{aligned}
\text{minimize} \quad & \textstyle\sum_{t=1}^{c} v_t a_t + \sum_{t=c+1}^{T} v_t \\
\text{subject to} \quad & \textstyle\sum_{t=1}^{T} v_t = 2\left(\sum_{t=1}^{c} v_t b_t + \sum_{t=c+1}^{T} v_t\right) \\
& \textstyle\sum_{t=1}^{T} v_t = \sum_{t=1}^{T} v_t\lambda_t \\
& \textstyle\sum_{t\in[T]:\lambda_t\le 1} v_t + \sum_{t\in[T]:\lambda_t>1} \lambda_t v_t = 1 \\
& v_t \ge 0, \text{ for all } t \in [T]
\end{aligned}
$$

where $a_t = \frac{1+(\lambda_t)^{p+1}}{1+(\lambda_t)^p}$ and $b_t = \frac{1}{1+(\lambda_t)^p}$.

By the fundamental theorem of linear programming, we must have T tight constraints, and we have $T+3$ total constraints (with the first three being tight), so any optimal solution should have exactly 3 strictly positive v_t variables.

We take cases depending on the value of c. Specifically, our three strictly positive v_t variables are either all three after the critical point, two and one, one and two, or all three before the critical point. The first case is, of course, impossible (since the first constraint cannot be satisfied), so we consider each of the other ones.

For each of the cases considered we write a closed form for the approximation to the welfare, as a function of the λ_ts, we then minimize. For $c=1$ (one item before, two items after the critical point) we get a worst-case approximation of 0.916. $c=3$, corresponding to no critical points, also gives a worst-case approximation. This corresponds to the intuition from Figure 1. Details can be found in the full version. □

We complement our positive result by showing that no fair-share adaptive algorithm, even with full knowledge of the number of items T, can achieve an approximation to the welfare much better than the guarded poly-proportional family. We defer the proof to the full version of this paper.

Theorem 4. *There is no fair-share algorithm that achieves an approximation to the optimal welfare better than 0.933.*

Instances Involving Multiple Agents

We now briefly turn to instances with $n \geq 3$. Caragiannis et al. (2012) prove that even if we knew all the values in advance, the *price of fairness*, i.e., the worst-case ratio of the optimal social welfare of a fair-share outcome over the social welfare of the optimal outcome, is $O(1/\sqrt{n})$. Our next result shows that the proportional algorithm matches this bound in an online manner, and therefore achieves the optimal approximation. We defer the proof to the full version.

Theorem 5. *The proportional algorithm guarantees a $\frac{1}{2\sqrt{n}}$, i.e., $\Omega(1/\sqrt{n})$, approximation to the optimal social welfare.*

The next result shows that even if we were to restrict the benchmark to be the optimal social welfare subject to the fair-share constraint, still, no online algorithm could achieve an approximation better than $\Omega(1/\sqrt{n})$. Therefore the proportional algorithm is also optimal with respect to the *competitive ratio* measure, which quantifies the worst-case loss of welfare due to the online aspect of the problem alone.

Theorem 6. *No online fair-share algorithm can achieve a $\frac{3\sqrt{n}}{n+\sqrt{n}-1}$ approximation to the optimal offline fair-share algorithm. That is, the best feasible approximation is $O(\frac{1}{\sqrt{n}})$.*

Proof. Consider an instance with n agents and $\sqrt{n}+n$ rounds. In the first $\sqrt{n}$ rounds, for the first $\sqrt{n}$ agents, we have $v_{ii} = \frac{n-1}{n}$, and $v_{it} = 0$, $t \neq i$. For the remaining $n-\sqrt{n}$ agents, we have $v_{jt} = \frac{n-1}{n\cdot\sqrt{n}}$, for all $j > \sqrt{n}$. Then, in the last n rounds, we have $v_{ii+\sqrt{n}} = \frac{1}{n}$, and $v_{it} = 0$ elsewhere, for all $i \in N$.

In the offline problem, each agent gets $\frac{1}{n}$ from the last n rounds. Therefore the optimal offline fair-share welfare is

$$OPT = \sqrt{n} \cdot \frac{n-1}{n} + n \cdot \frac{1}{n} = \frac{n-1}{\sqrt{n}} + 1.$$

We now focus our attention on round $\sqrt{n}$. Note that each agent has remaining value $1/n$ at this round. An online fair-share algorithm needs to plan for the event that the remaining values are all realized in the next round, $\sqrt{n}+1$. In order to satisfy fair-share in this scenario, each agent must have utility at least $\frac{n-1}{n}\frac{1}{n} = \frac{n-1}{n^2}$ at the end of round $\sqrt{n}$.

Consider an agent i with $i > \sqrt{n}$. Since her value for all the items before round $\sqrt{n}$ is $v_{it} = \frac{n-1}{n\sqrt{n}}$, to give this agent utility at least $\frac{n-1}{n^2}$ her total allocation must be $\sum_{t=1}^{\sqrt{n}} x_{it} \geq \frac{n-1}{n^2}\frac{n\sqrt{n}}{n-1} = \frac{1}{\sqrt{n}}$. This is true for all $i > \sqrt{n}$, so there is $\sqrt{n} - (n-\sqrt{n})\frac{1}{\sqrt{n}} = 1$ of the resources, in the first $\sqrt{n}$, to be allocated among the first $\sqrt{n}$ agents. No matter how this is split, the contribution to the welfare is the same. Let U^t be the social welfare at the end of round t. We have

$$U^{\sqrt{n}} = 1 \cdot \frac{n-1}{n} + (n-\sqrt{n})\frac{n-1}{n^2} = 2 - \frac{2\sqrt{n}+n+1}{n\sqrt{n}}.$$

For the last n rounds our algorithm can make an optimal choice: $ALG = U^{\sqrt{n}} + n \cdot \frac{1}{n} = 3 - \frac{2\sqrt{n}+n+1}{n\sqrt{n}} < 3$. Therefore, we have $\alpha = \frac{ALG}{OPT} < \frac{3}{\frac{n-1}{\sqrt{n}}+1} = \frac{3\sqrt{n}}{n+\sqrt{n}-1}$. □

Characterization of Fair-Share Algorithms

Our final result provides an interesting characterization of fair-share algorithms that could enable the design of novel algorithms in this setting. This characterization uses a very simple condition, which we refer to as doomsday compatibility, and we show that this myopic condition is necessary, but also sufficient, for guaranteeing that the final outcome will satisfy fair-share.

Definition 1 (Doomsday Compatibility)**.** *We say an allocation $\mathbf{x}^t = \{x_{it}\}_{i\in N}$ at day t is* doomsday compatible *if there exists some allocation $\mathbf{x}^{t+1}$ that would make the overall outcome satisfy fair-share, if $t+1$ was the last round, i.e., if all the agents' remaining value was realized in round $t+1$.*

Proposition 1. *An online algorithm satisfies the fair-share property if and only if its allocation in every round t is doomsday compatible.*

Proof. First, it is easy to show that doomsday compatibility in every round t is sufficient for an online algorithm to satisfy fair-share. If this condition is satisfied for all t, then it is also satisfied for $t = T-1$ and $t = T$, and thus the final outcome is guaranteed to satisfy fair-share.

Now, we show that this condition is also necessary for the algorithm to satisfy fair-share. Assume that there exists a round t such that the online algorithm's allocation in this round is not doomsday compatible. Then, clearly this algorithm would not be fair-share for the instance where $t+1$ is indeed the last round, i.e., where all of the agents' remaining value is realized in round $t+1$. □

Theorem 7. *If an algorithm is doomsday-compatible in some round $t < T$, then there always exists an allocation $\mathbf{x}^{t+1}$ such that it is also doomsday compatible in round $t+1$.*

Proof. Consider any round t where the algorithm's allocation is doomsday compatible. This means that there exists some allocation $\tilde{\mathbf{x}}$ that would achieve fair-share if $t+1$ was the last round. In order to show that we can always maintain doomsday compatibility in round $t+1$, it suffices to show that there always exists some allocation $\mathbf{x}^{t+1}$ for that round and an allocation $\mathbf{x}^{t+2}$ for the next round such that the algorithm would satisfy fair-share if $t+2$ were the last round. We show that, in fact, using $\tilde{\mathbf{x}}$ for both rounds $t+1$ and $t+2$ would satisfy this condition.

To verify this fact, let $\bar{\mathbf{v}}_{it}$ be the remaining value for each agent i after round t, and let u_i be the total utility each agent received up to round t. Since $\tilde{\mathbf{x}}$ would make the outcome fair-share if $t+1$ was the last round, for any agent i we have $u_i + \bar{\mathbf{v}}_{it}\tilde{\mathbf{x}} \geq \frac{1}{n}$. Now, if on the other hand $t+2$ was the last round, let $\mathbf{x}^{t+1} = \tilde{\mathbf{x}}$ and $\mathbf{x}^{t+2} = \tilde{\mathbf{x}}$. Then, for any agent i we would have

$$\begin{aligned} &u_i + v_{i(t+1)}\mathbf{x}^{t+1} + (\bar{\mathbf{v}}_{it} - v_{i(t+1)})\mathbf{x}^{t+2} \\ =&u_i + v_{i(t+1)}\tilde{\mathbf{x}} + (\bar{\mathbf{v}}_{it} - v_{i(t+1)})\tilde{\mathbf{x}} \\ =&u_i + \bar{\mathbf{v}}_{it}\tilde{\mathbf{x}} \geq \frac{1}{n}. \end{aligned}$$

Therefore, for $\mathbf{x}^{t+1} = \tilde{\mathbf{x}}$, there exists a $\mathbf{x}^{t+2} = \tilde{\mathbf{x}}$ such that the algorithm is doomsday compatible in round $t+1$. □

Acknowledgments

Part of this work took place when Vasilis Gkatzelis and Alexandros Psomas were visiting the Simons Institute for the Theory of Computing. Also, part of this work was done while Alexandros Psomas was at Google Research, Mountain View. This work was partially supported by NSF grants CCF-1755955 and CCF-2008280.

References

Aziz, H. 2019. Justifications of welfare guarantees under normalized utilities. *SIGecom Exch.* 17(2): 71–75.

Benade, G.; Kazachkov, A. M.; Procaccia, A. D.; and Psomas, C.-A. 2018. How to make envy vanish over time. In *Proceedings of the 2018 ACM Conference on Economics and Computation*, 593–610.

Bogomolnaia, A.; Moulin, H.; and Sandomirskiy, F. 2019. A simple Online Fair Division problem. *CoRR* abs/1903.10361.

Brânzei, S.; Gkatzelis, V.; and Mehta, R. 2017. Nash Social Welfare Approximation for Strategic Agents. In Daskalakis, C.; Babaioff, M.; and Moulin, H., eds., *Proceedings of the 2017 ACM Conference on Economics and Computation, EC '17, Cambridge, MA, USA, June 26-30, 2017*, 611–628. ACM.

Caragiannis, I.; Kaklamanis, C.; Kanellopoulos, P.; and Kyropoulou, M. 2012. The Efficiency of Fair Division. *Theory Comput. Syst.* 50(4): 589–610.

Christodoulou, G.; Sgouritsa, A.; and Tang, B. 2016. On the Efficiency of the Proportional Allocation Mechanism for Divisible Resources. *Theory Comput. Syst.* 59(4): 600–618.

Feldman, M.; Lai, K.; and Zhang, L. 2009. The Proportional-Share Allocation Market for Computational Resources. *IEEE Transactions on Parallel and Distributed Systems* .

Friedman, E.; Psomas, C.-A.; and Vardi, S. 2015. Dynamic fair division with minimal disruptions. In *Proceedings of the sixteenth ACM conference on Economics and Computation*, 697–713.

Friedman, E.; Psomas, C.-A.; and Vardi, S. 2017. Controlled dynamic fair division. In *Proceedings of the 2017 ACM Conference on Economics and Computation*, 461–478.

Gorokh, A.; Banerjee, S.; Jin, B.; and Gkatzelis, V. 2020. Online Nash Social Welfare via Promised Utilities. *CoRR* abs/2008.03564.

He, J.; Procaccia, A. D.; Psomas, A.; and Zeng, D. 2019. Achieving a fairer future by changing the past. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, 343–349. AAAI Press.

Im, S.; Moseley, B.; Munagala, K.; and Pruhs, K. 2020. Dynamic Weighted Fairness with Minimal Disruptions. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 4(1): 1–18.

Kash, I. A.; Procaccia, A. D.; and Shah, N. 2014. No Agent Left Behind: Dynamic Fair Division of Multiple Resources. *J. Artif. Intell. Res.* 51: 579–603.

Prendergast, C. 2017. How Food Banks Use Markets to Feed the Poor. *Journal of Economic Perspectives* 31(4).

Walsh, T. 2011. Online cake cutting. In *International Conference on Algorithmic DecisionTheory*, 292–305. Springer.

Zeng, D.; and Psomas, A. 2020. Fairness-Efficiency Tradeoffs in Dynamic Fair Division. In Biró, P.; Hartline, J.; Ostrovsky, M.; and Procaccia, A. D., eds., *EC '20: The 21st ACM Conference on Economics and Computation, Virtual Event, Hungary, July 13-17, 2020*, 911–912. ACM.

Zhang, L. 2005. The Efficiency and Fairness of a Fixed Budget Resource Allocation Game. In Caires, L.; Italiano, G. F.; Monteiro, L.; Palamidessi, C.; and Yung, M., eds., *Automata, Languages and Programming, 32nd International Colloquium, ICALP 2005, Lisbon, Portugal, July 11-15, 2005, Proceedings*, volume 3580 of *Lecture Notes in Computer Science*, 485–496. Springer.